AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated in the following listing of all claims:

1-8. Cancelled

9. (currently amended) A superscalar processor that, for a given instruction instance, performs, over plural execution cycles of the superscalar processor, instruction grouping for dispatch, including both intra-group and inter-group dependency checking.

10. (previously presented) The superscalar processor of claim 9, further comprising:

grouping logic implementing plural early pipeline stages of the superscalar processor; and

plural execution units of varying pipeline depth coupled to receive instructions dispatched from the grouping logic.

11. (previously presented) The superscalar processor of claim 9,

wherein the instruction grouping identifies successive groups of instructions from an instruction stream for dispatch to respective ones of plural execution units of the superscalar processor.

12. (previously presented) The superscalar processor of claim 11,

wherein the superscalar processor dispatches all instructions from a particular one of the successive groups before dispatching any instructions from a subsequent one of the successive groups.

13. (previously presented) The superscalar processor of claim 9,

wherein the intra-group dependency checking spans at least two of the plural execution cycles.

14. (previously presented) The superscalar processor of claim 9,

wherein the intra-group dependency checking is independent of the inter-group dependency checking.

15. (previously presented) The superscalar processor of claim 9,
wherein data dependency and resource allocation checks in earlier pipeline stages of instruction grouping are based, at least in part, on a predicted subsequent state of the superscalar processor.

16. (previously presented) The superscalar processor of claim 9,
wherein non-deterministic conditions are evaluated in a final stage of instruction grouping prior to dispatch.

17. (previously presented) A processor comprising:
plural functional units that execute instructions in respective numbers of processor cycles; and
grouping logic coupled to the functional units and pipelined to compute, over plural cycles, T, of the processor, a future state, S(t+T), of the processor based on a prior state, S(t), of the processor and based thereon to select a group of instructions from a program sequence thereof for dispatch to the functional units.

18. (previously presented) The processor of claim 17, further comprising:
wherein intragroup dependency checking by the pipelined grouping logic spans two or more of the T cycles.

19. (previously presented) The processor of claim 17, further comprising:
wherein inter-group dependency checking is performed independently of intra-group dependency checking.

20. (previously presented) The processor of claim 17,
wherein intra-group dependencies are checked by the pipelined grouping logic beginning in a first of the T cycles; and
wherein non-deterministic dependency conditions are checked during a last of the T cycles.

21. (previously presented) The processor of claim 20,

wherein inter-group dependencies are checked independent of the intra-group dependencies.

22. (previously presented) The processor of claim 17,
wherein the grouping logic implements T stages of a pipeline of the processor.

23. (previously presented) The processor of claim 17,
wherein T is four.

24. (previously presented) The processor of claim 17,
wherein the functional units include at least one functional unit capable of receiving and completing an instruction for each of the processor cycles.

25. (previously presented) The processor of claim 17,
wherein the functional units include at least one functional unit requiring multiple of the processor cycles for receiving and completing an instruction.

26. (previously presented) A method of operating a processor, the method comprising:
identifying successive groups of instructions for dispatch to respective ones of plural execution units of the processor;
performing, during plural pipelined execution cycles of the processor, dependency checking amongst instructions of a later one of the groups and between the instructions of the later group and instructions of a preceding one of the groups; and
dispatching instructions of the later group only after all instructions of the preceding group have been dispatched.

27. (previously presented) The method of claim 26,
wherein the dependency checking is performed by pipelined grouping logic.

28. (previously presented) The method of claim 26,

wherein intra-group dependency checking and within group resource allocation are performed in successive processor cycles by pipelined grouping logic.

29. (previously presented) The method of claim 26,
wherein intra-group dependency checking is performed independent of inter-group dependency checking.

30. (previously presented) The method of claim 26,
wherein non-deterministic conditions are evaluated in a final one of the pipelined execution cycles implemented by pipelined grouping logic.

31. (previously presented) The method of grouping instructions for dispatch to execution units of a processor, the method comprising:
in a first cycle of processor execution, identifying plural candidate instructions for an instruction group;
in a subsequent cycle of processor execution, beginning intra-group dependency checking as amongst instructions of the instruction group;
in a cycle of processor execution prior to dispatch of any instruction from the instruction group, checking non-deterministic conditions; and
in a cycle of processor execution prior to the non-deterministic dependency condition checking, initiating inter-group dependency checking as between instructions of the instruction group and instructions of one or more prior instruction groups.

32. (previously presented) The method of claim 31, further comprising:
dispatching one or more instructions of the instruction group only after all instructions of the one or more prior instruction groups have been dispatched.

33. (previously presented) The method of claim 31,
wherein the non-deterministic condition checking is performed conservatively, based on a assumption of no change in condition.

34. (previously presented) The method of claim 31,

wherein the non-deterministic condition checking is performed aggressively, computing dispatch conditions for at least two alternatives including no change in condition and condition resolution; and

wherein a control signal is selective for a particular one of the computed alternatives.

35. (previously presented) An apparatus comprising:

plural functional units; and

means for grouping, over plural pipeline stages, instructions for dispatch to respective ones of the functional units.

36. (previously presented) The apparatus of claim 35, wherein the grouping means includes:

means for deriving over T processor cycles, a future state S(t+T) based on a present state; and

means for determining a group of the instructions to dispatch at time t+T.